Ada®
FROM A MANAGEMENT PERSPECTIVE

by: Major Charles Engle
1Lt Anthony D. Dominice

Presented by: Ada® Software Engineering Education & Training (ASEET) Team

Sponsored by: Ada® Joint Program Office

Organized by: Herbert E. Cohen
U.S. Army Materiel Systems Analysis Activity
Aberdeen Proving Ground, Md.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION AMSAA 6b. OFFICE SYMBOL (If applicable) AMXSY-MP	7a. NAME OF MONITORING ORGANIZATION Same as 6
6c. ADDRESS (City, State, and ZIP Code) ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005-5071	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Ada Joint Program Office (OCD) 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) The Pentagon Washington, D.C. 20301-3081	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Ada from a Management Perspective

12. PERSONAL AUTHOR(S)
Herb Cohen, Organizer

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Fianl	FROM ______ TO ______	88/3/31	81

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Engineering, Abstraction, Modularity, Localization Information Hiding, Packages, Subprograms, Generics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Provides an introductory review of principles of software engineeirng and major features of ADA.

95-01573

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Herbert E. Cohen	22b. TELEPHONE (Include Area Code) (301)267-2785/6577 22c. OFFICE SYMBOL AMSAA(AMXSY-MP)

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Request for Video Tapes

1. DOD and other government agencies can obtain copies of tapes and text through their nearest local Training and Audio Visual Support Center.

2. Reference tape PIN # 505193.

3. Local Training and Audio Visual Support Center can request tapes through:

   Department of the Army
   U.S. Army Visual Information Center
   Joint Visual Information Activity
   ATTN: ASNV-OJVP-CM
   Tobyhanna Army Depot, PA 18466-5102

Tapes will be standard DOD, 3/4 inch video cassette; however 1/2 inch VHS are also available on request.

4. The general public can obtain tapes at minimal cost, in any of the formats specified above, by writing to:

   National Audio Visual Center
   GSA
   ATTN: Order Section
   Washington, D.C. 20409

5. For additional information, contact:

   Ada Joint Program Office
   Pentagon 3E114
   Washington, D.C. 20301-3080
   PHONE: (202) 694-0209

ACKNOWLEDGEMENT

The U.S. Army Materiel Systems Analysis Activity (AMSAA) wishes to express its deep appreciation to Major Charles Engle (ARMY) and 1st Lieutenant Anthony D. Dominice (AF) of the Ada Software Engineering Education and Training (ASEET) Team for an outstanding lecture series. This activity could not have been carried out without the continuing support of Lieutenant Colonel David Taylor (ARMY) and Major Allen Kopp (AF) of the Ada Joint Program Office (OSD). This video production required considerable effort and professionalism by the Combat Systems Test Activity at Aberdeen Proving Ground, Maryland in particular, Jack Frost, Dave Jennings and Larry Ross which is deeply appreciated.

Overview

Rationale for development

Capabilities and advantages

Life Cycle application

What you may have heard about Ada

It's a cure all for DoD computing problems

It's just another acronym

It's a programming language

It's "just another" programming language

It's a panacea

What you need to hear about Ada

Plain and simple...

Ada is a standardized computer language developed by the DoD for use in embedded computer systems

Ada is the BEST tool available for meeting the software engineering requirements of the DoD

Overview

Rationale for development

Capabilities and advantages

Life Cycle application

Overview

Rationale for development

Capabilities and advantages

Life Cycle application

The criticality of software

Hardware is no longer the dominant factor in the hardware/software relationship

The demand for software is rising quickly

The cost of software is rising quickly

Software maintenance is the dominant software activity

Systems are getting more complex

Life and property are dependent on software

Characteristics of DoD software

Expensive

Incorrect

Unreliable

Unpredictable

Unmaintainable

Not reusable

Factors affecting DoD software

Ignorance of life cycle implications

Lack of standards

Lack of disciplined use of methodologies

Inadequate support environments

Management

Software professionals

Characteristics of DoD software requirements

Large

Complex

Long lived

High reliability

Time constraints

Size constraints

The fundamental problem

Our inability to manage the COMPLEXITY of our software systems (Grady Booch)

Lack of a disciplined, engineering approach

Software Engineering

The establishment and application of sound engineering

Environments

Tools

Methodologies

Models

Principles

Concepts

Software Engineering

Combined with

Standards

Guidelines

Practices

Software Engineering

To support computing which is

Understandable

Efficient

Reliable and safe

Modifiable

Correct

Throughout the life cycle of a system

(C. McKay, 1985)

Programming languages and software engineering

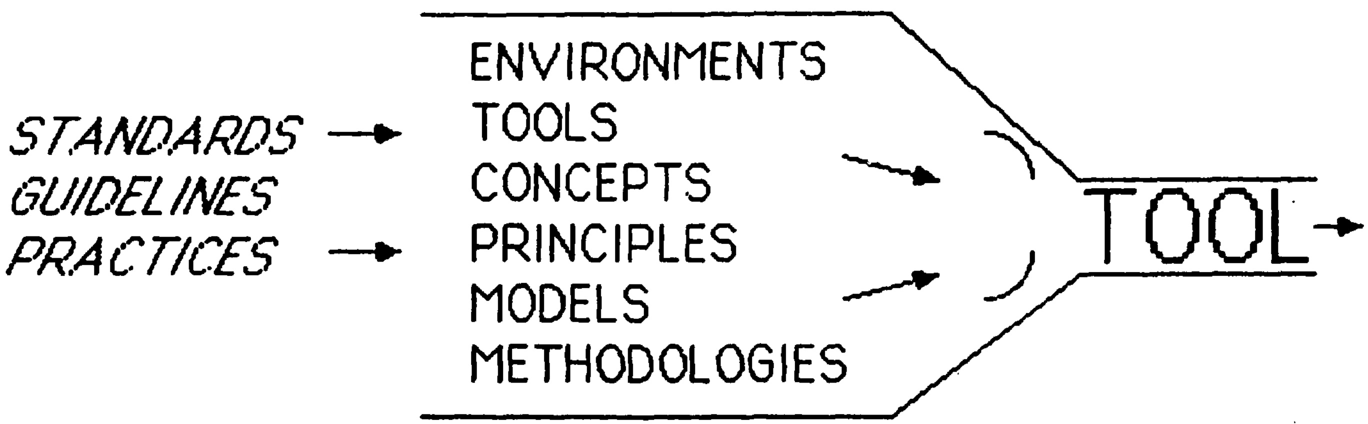
A programming language is a software engineering tool

A programming language EXPRESSES and EXECUTES design methodologies

The quality of a programming language for software engineering is determined by how well it supports a design methodology and its underlying models, principles, and concepts

Traditional programming languages and software engineering

Programming Languages	Were not engineered
	Have lacked the ability to express good software engineering
	Have acted to constrain software engineering



Ada
and
software engineering

Ada — Was itself engineered to support software engineering

Embodies the same concepts, principles, and models to support methodologies

Is the best tool for software engineering available

STANDARDS
GUIDELINES
PRACTICES

→

ENVIRONMENTS
TOOLS
CONCEPTS
PRINCIPLES
MODELS
METHODOLOGIES

→ TOOL →

Principles of software engineering

Abstraction

Modularity

Localization

Information hiding

Completeness

Confirmability

Uniformity

Abstraction

The process of separating out the important parts of something while ignoring the inessential details

Separates the "what" from the "how"

Reduces the level of complexity

There are levels of abstraction within a system

Modularity

Purposeful structuring of a system into parts which work together

Each part performs some smaller task of the overall system

Can concentrate and develop parts independently as long as interfaces are defined and shared

Can develop hierarchies of management and implementation

Localization

Putting things that logically belong together in the same physical place

Information hiding

Puts a wall around localized details

Prevents reliance upon details and causes focus of attention to interfaces and logical properties.

Completeness

Ensuring all important parts are present

Nothing left out

Confirmability

Developing parts that can be effectively tested

Uniformity

No unnecessary differences across a system

Major Features Of Ada

Standardization

Readability

Program Units

Separate Compilation

Subprograms

Packages

Strong Typing

Typing Structures

Data Abstraction

Tasks

Exceptions

Generics

Major Features Of Ada

- Standardization
- Readability
- Program Units
- Separate Compilation
- Subprograms
- Packages
- Strong Typing
- Typing Structures
- Data Abstraction
- Tasks
- Exceptions
- Generics

Standardization

Ada is an exact standard

- ANSI/MIL STD 1815A
- No subsets, no supersets

Conformance to the standard is required

- Ada Compiler Validation Capability (ACVC)

Standardization allows for portability

Standardization promotes reusability

Standardization shifts focus from the mundane to the important

Major Features Of Ada

Standardization	Strong Typing
Readability	Typing Structures
Program Units	Data Abstraction
Separate Compilation	Tasks
Subprograms	Exceptions
Packages	Generics

Readability

Ada was engineered with the understanding that programming is a human activity

Features are provided that allow a maintainer to quickly grasp the meaning of a particular program and to understand its structure

Readability is more than just a language issue

Major Features Of Ada

Standardization

Readability

Program Units

Separate Compilation

Subprograms

Packages

Strong Typing

Typing Structures

Data Abstraction

Tasks

Exceptions

Generics

Systems engineering

Analyze problem

Break into solvable parts

Implement parts

Test parts

Integrate parts to form total system

Test total system

Requirements for effective systems engineering

Ability to express architecture

Ability to define and enforce interfaces

Ability to create independent components

Ability to separate architecture issues from implementation issues

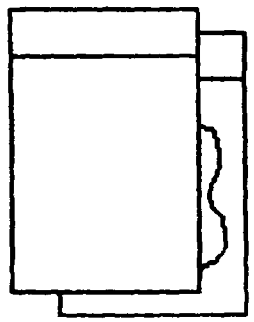
Program Units

Components of Ada which together form a working Ada software system

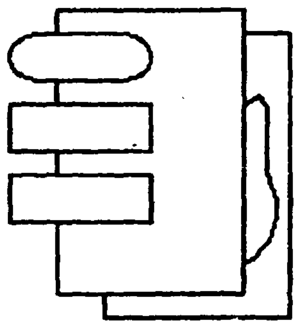
Express the architecture of a system

Define and enforce interfaces

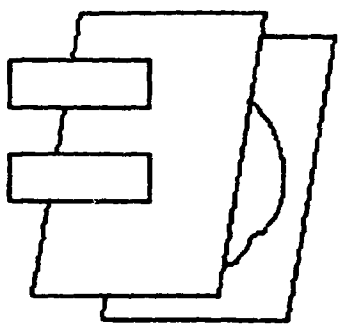
Program Units



Subprograms — Working components that perform some action



Packages — A mechanism for collecting logically related entities

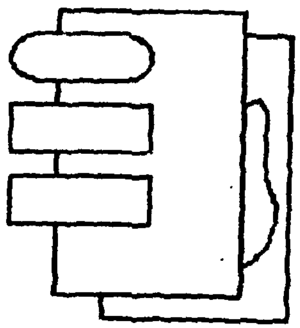


Tasks — Perform actions in parallel with other entities

Program Units

Consist of two parts: specification and body

SPECIFICATION:
Defines the interface between the unit and other units (the WHAT)



BODY: Defines the implementation of the unit (the HOW)

Program Units

The specification of program units is the only means of connecting them together

The interface is enforced and shared

The body of a program unit is not accesible to other program units

There is a clear distinction between architecture and implementation

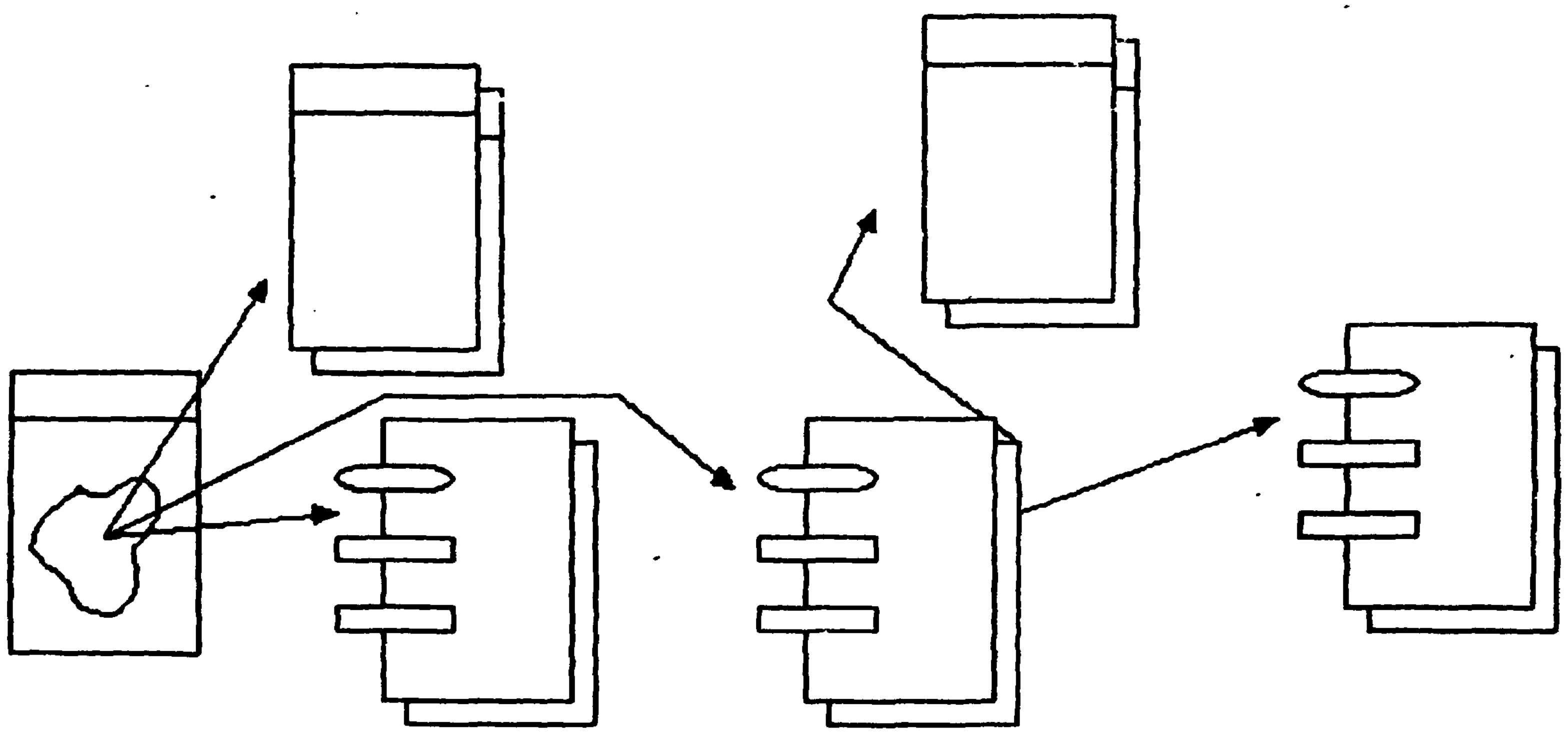
Major Features Of Ada

- Standardization
- Readability
- Program Units
- $\boxed{\text{Separate Compilation}}$
- Subprograms
- Packages
- Strong Typing
- Typing Structures
- Data Abstraction
- Tasks
- Exceptions
- Generics

Separate compilation

Program units may be separately compiled

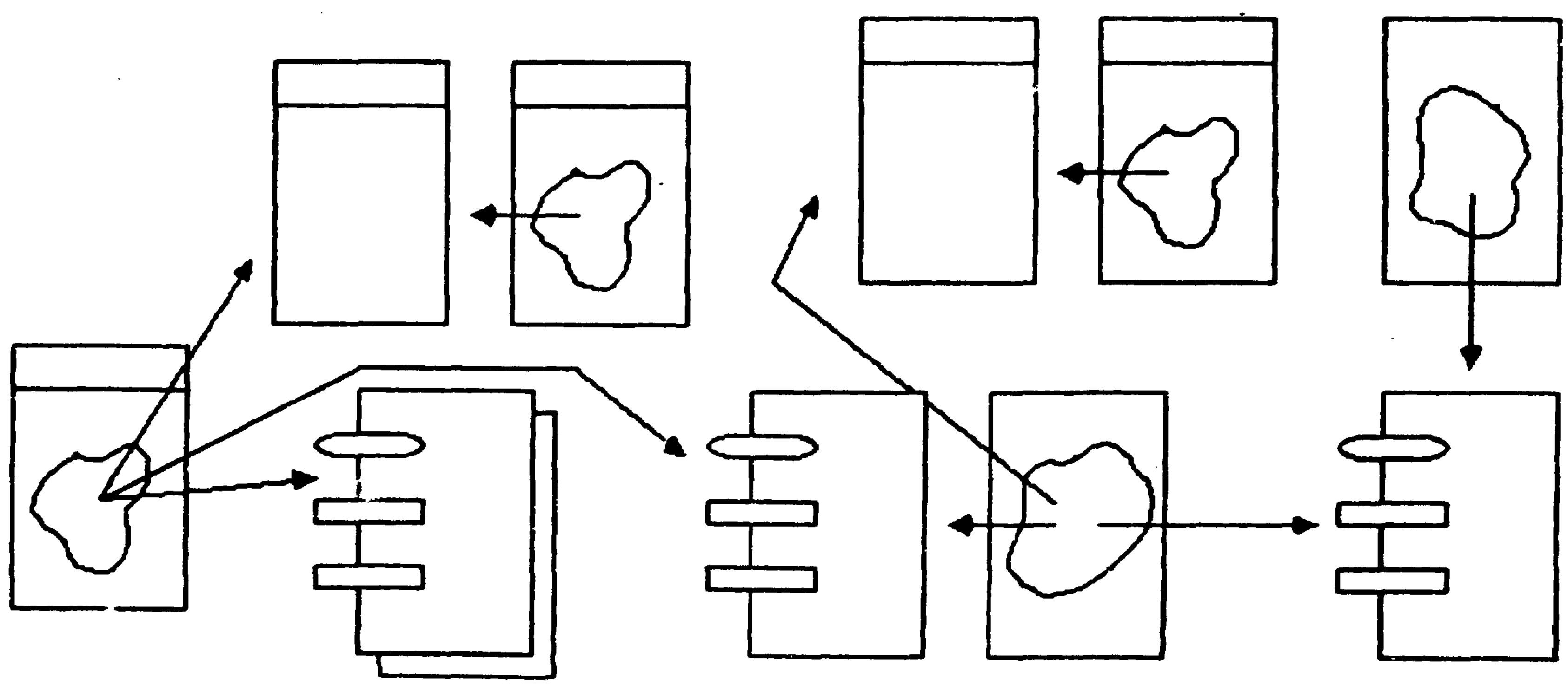
Possible because of separation of specification and body



Separate compilation

A program unit's spec may be compiled separately from it's body

Realizes not only a logical distinction between architecture and implementation, but also a physical distinction



Separate compilation

Allows development of independent software components

Currently we lose a tremendous amount of human effort; software is disposable

Separate compilation allows us to reuse components and keep our investment

Major Features Of Ada

Standardization

Readability

Program Units

Separate Compilation

Subprograms

Packages

Strong Typing

Typing Structures

Data Abstraction

Tasks

Exceptions

Generics

Discrete components

Allow a system to be composed of black boxes

Provide clear understandable functions

Can be verified and validated more effectively

Prevalent across engineering disciplines

Subprograms

A program unit that performs a particular action

- Procedures
- Functions

Contains an interface (parameter part) mechanism to pass data to and from the subprogram

The basic discrete component which acts like a black box

Gives ability to express abstract actions

Major Features Of Ada

- Standardization
- Readability
- Program Units
- Separate Compilation
- Subprograms
- $\boxed{\text{Packages}}$
- Strong Typing
- Typing Structures
- Data Abstraction
- Tasks
- Exceptions
- Generics

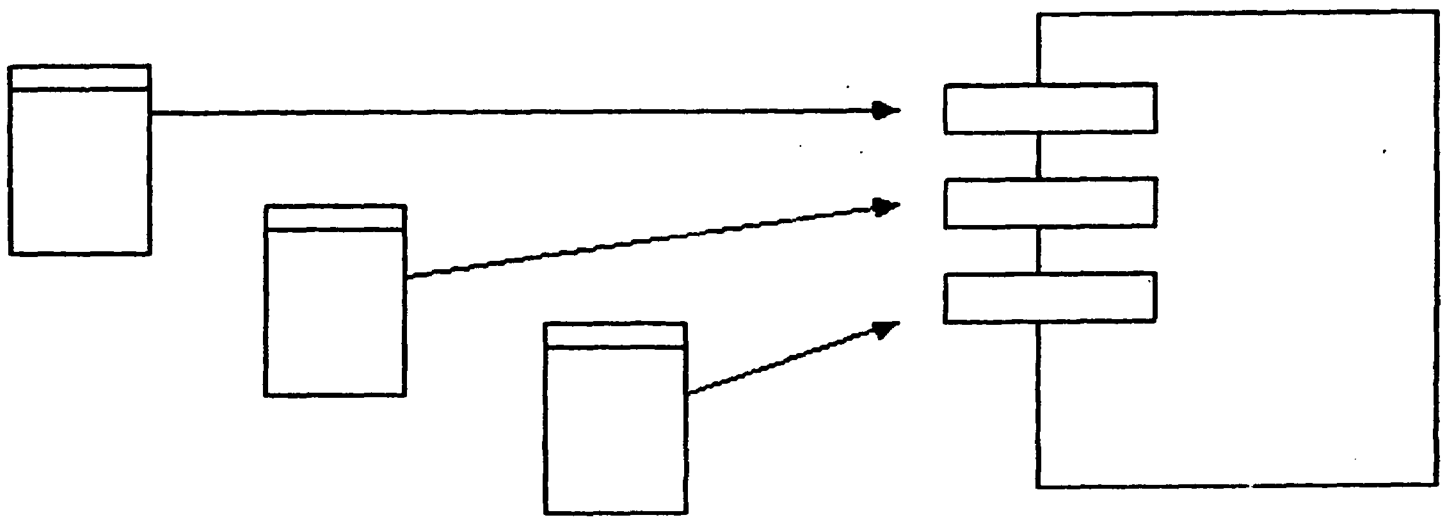
Packages

Program units that allow us to collect logically related entities

Allow the definition of reuseable software components

A fundamental feature of Ada which allows a change of mindset

An architecture oriented feature

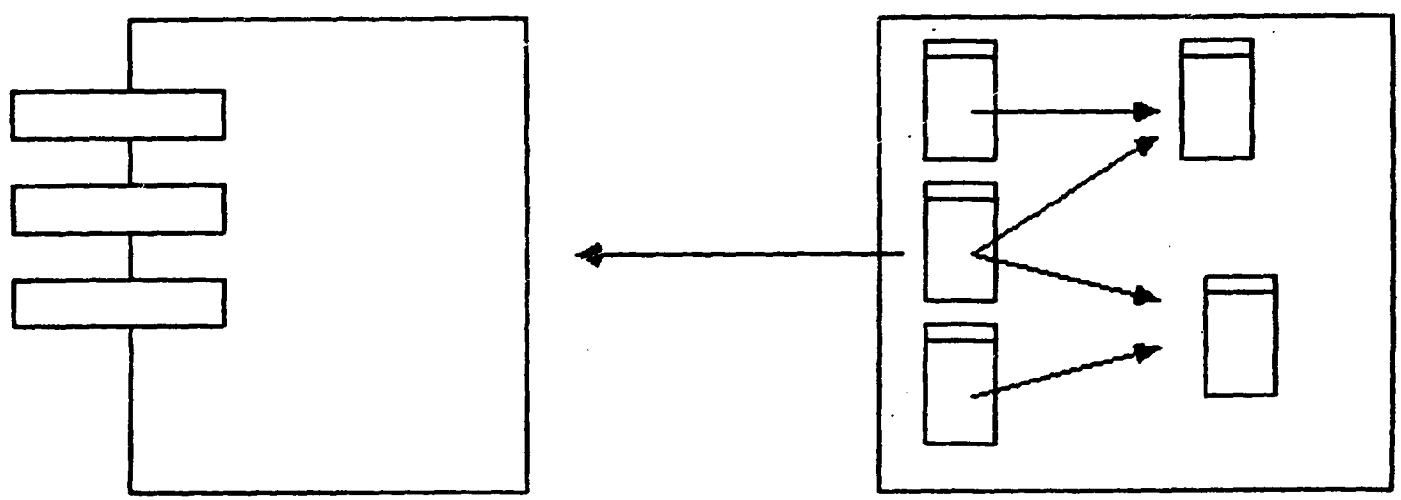


Packages

Place a wall around resources

Export resources to users of a package

May contain local resources hidden from the user



Packages

Directly support:

Abstraction

Information hiding

Modularity

Localization

Understandability, efficiency, reliability, modifiability

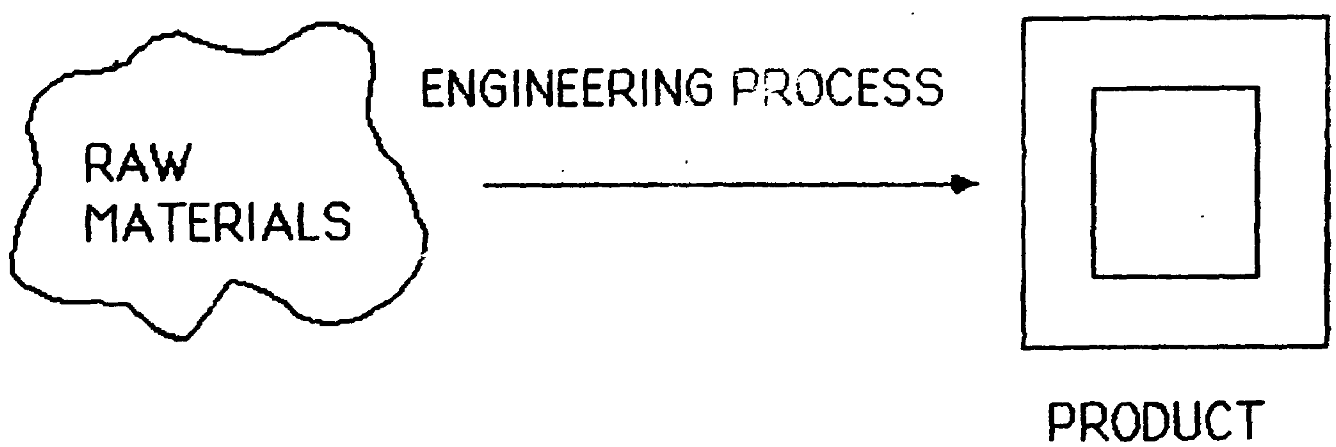
Major Features Of Ada

Standardization	Strong Typing
Readability	Typing Structures
Program Units	Data Abstraction
Separate Compilation	Tasks
Subprograms	Exceptions
Packages	Generics

The raw materials of engineering

All engineering disciplines shape raw materials into a finished product

The materials and methods combine to define different disciplines



Structuring raw materials

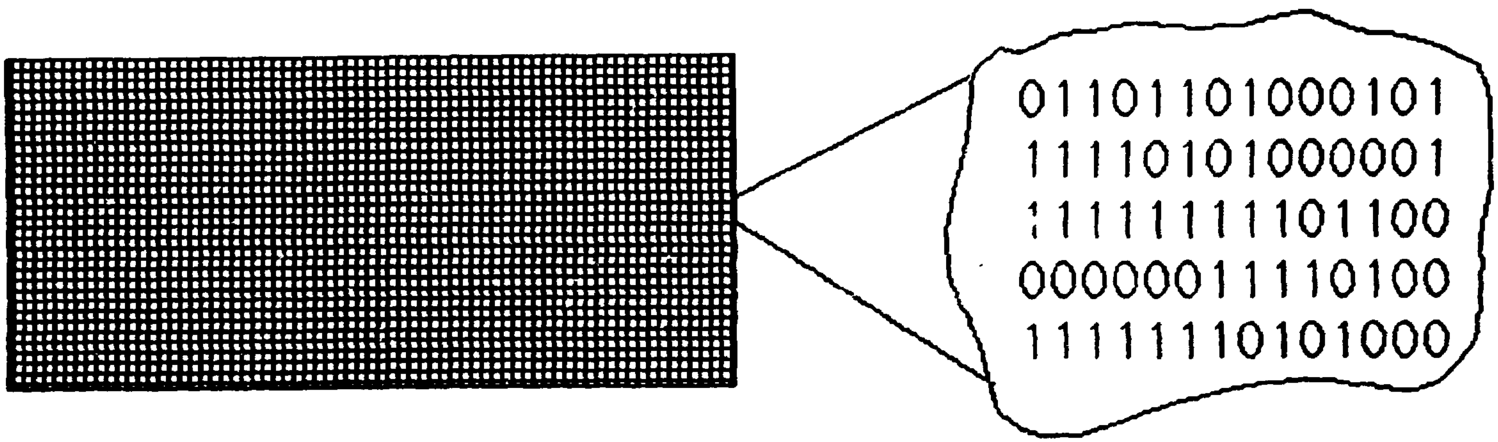
There is a requirement to structure raw materials

- To quantify
- To manage
- To test
- To validate

Methods of structuring vary across disciplines

Some raw materials of software engineering

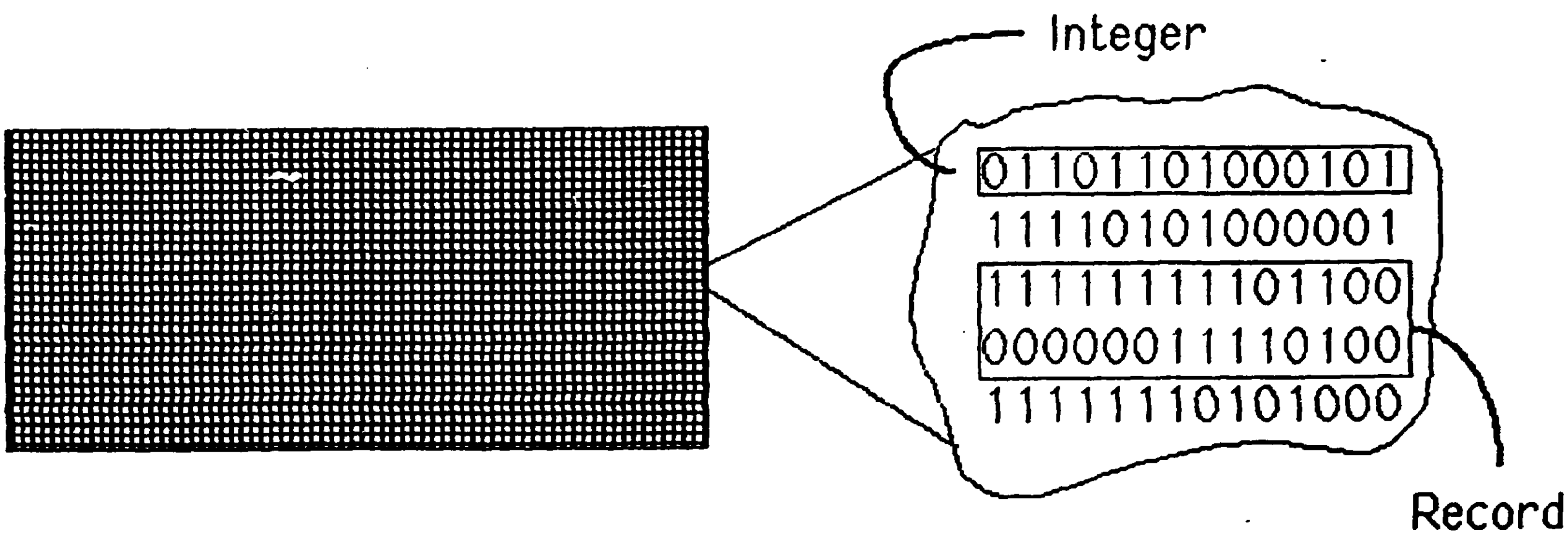
Binary switches
Computer memory locations
Data



Strong typing

Defines structure of data

Enforces structure of data



Strong typing

Enforces abstraction of structure on data

Increases confidence of correctness

Increases reliability and safety

Promotes understandability and maintainability

Major Features Of Ada

Standardization	Strong Typing
Readability	$\boxed{\text{Typing Structures}}$
Program Units	Data Abstraction
Separate Compilation	Tasks
Subprograms	Exceptions
Packages	Generics

Typing Structures

Variety of problems requires a variety of structuring capabilities

Ada provides a rich variety of typing capabilities

Typing structures in Ada

Discrete data
- Enumerated
- Integer

Real data
- Fixed point (absolute error)
- Floating point (relative error)

Composite data
- Arrays (homogeneous)
- Records (heterogeneous)

Dynamic data
- Access types

Major Features Of Ada

Standardization	Strong Typing
Readability	Typing Structures
Program Units	Data Abstraction
Separate Compilation	Tasks
Subprograms	Exceptions
Packages	Generics

Data Abstraction

Combines primitive raw materials to form higher level structures

Levels of abstraction

Enforces an abstraction on a higher level structure

Prohibits use of implementation details

Promotes understandability

Promotes modifiability

Data abstraction and private types

Private types directly implement data abstraction

Directly implement information hiding

```
package B_R is

    type NUMBERS is range 0 .. 99;

    procedure TAKE ( A_NUMBER : out NUMBERS );

    procedure SERVE ( NUMBER : in NUMBERS );

    function NOW_SERVING return NUMBERS;

end B_R;
```

```
with B_R;
use   B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;

begin

   TAKE ( YOUR_NUMBER );

   loop

      if NOW_SERVING = YOUR_NUMBER then
         SERVE ( YOUR_NUMBER );
         exit;
      end if;

   end loop;

end ICE_CREAM;
```

```
with B_R;
use  B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;

begin

   TAKE ( YOUR_NUMBER );

   loop
      if NOW_SERVING = YOUR_NUMBER then
         SERVE ( YOUR_NUMBER );
         exit;
      else
         YOUR_NUMBER := YOUR_NUMBER - 1;
      end if;
   end loop;

end ICE_CREAM;
```

```
package B_R is

    type NUMBERS is private;

    procedure TAKE ( A_NUMBER : out NUMBERS );

    procedure SERVE ( NUMBER : in NUMBERS );

    function NOW_SERVING return NUMBERS;

private

    type NUMBERS is range 0 ..99;

end B_R;
```

```
with B_R;
use  B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;

begin

   TAKE ( YOUR_NUMBER );

   loop
      if NOW_SERVING = YOUR_NUMBER then
         SERVE ( YOUR_NUMBER );
         exit;
      else
         YOUR_NUMBER := NOW_SERVING;
      end if;
   end loop;

end ICE_CREAM;
```

```
package B_R is
   type NUMBERS is limited private;
   procedure TAKE ( A_NUMBER : out NUMBERS );
   procedure SERVE ( NUMBER : in NUMBERS );
   function NOW_SERVING return NUMBERS;
   function "=" ( LEFT, RIGHT : in NUMBERS ) return BOOLEAN;
private
   type NUMBERS is range 0 ..99;
end B_R;
```

```
with B_R;
use  B_R;
procedure ICE_CREAM is

   YOUR_NUMBER : NUMBERS;
   procedure GO_TO_DQ is separate;

begin

   TAKE ( YOUR_NUMBER );

   loop
      if NOW_SERVING = YOUR_NUMBER then
         SERVE ( YOUR_NUMBER );
         exit;
      else
         GO_TO_DQ;
      end if;
   end loop;

end ICE_CREAM;
```

Major Features Of Ada

Standardization	Strong Typing
Readability	Typing Structures
Program Units	Data Abstraction
Separate Compilation	$\boxed{\text{Tasks}}$
Subprograms	Exceptions
Packages	Generics

Tasks

Program unit that acts in parallel with other entities

Directly implements those parts of embedded systems which act in parallel

Takes advantage of move toward parallel hardware architectures

- Fault tolerance
- Distributed systems

Eliminates need to introduce additional complexity into a system

Major Features Of Ada

Standardization	Strong Typing
Readability	Typing Structures
Program Units	Data Abstraction
Separate Compilation	Tasks
Subprograms	Exceptions
Packages	Generics

Software reliability and safety

Errors will occur
- Hardware
- Software

Real time systems must be able to operate in a degraded mode

Reliability and safety must be engineered into a system

Traditional languages lack features for dealing with errors and exceptional situations

Exceptions

Deal specifically with errors and exceptional situations

When an exception is raised processing is suspended and control is passed to an appropriate exceptional handler

- Try again
- Fix error
- Propogate exception

Increases reliability

Reduces complexity

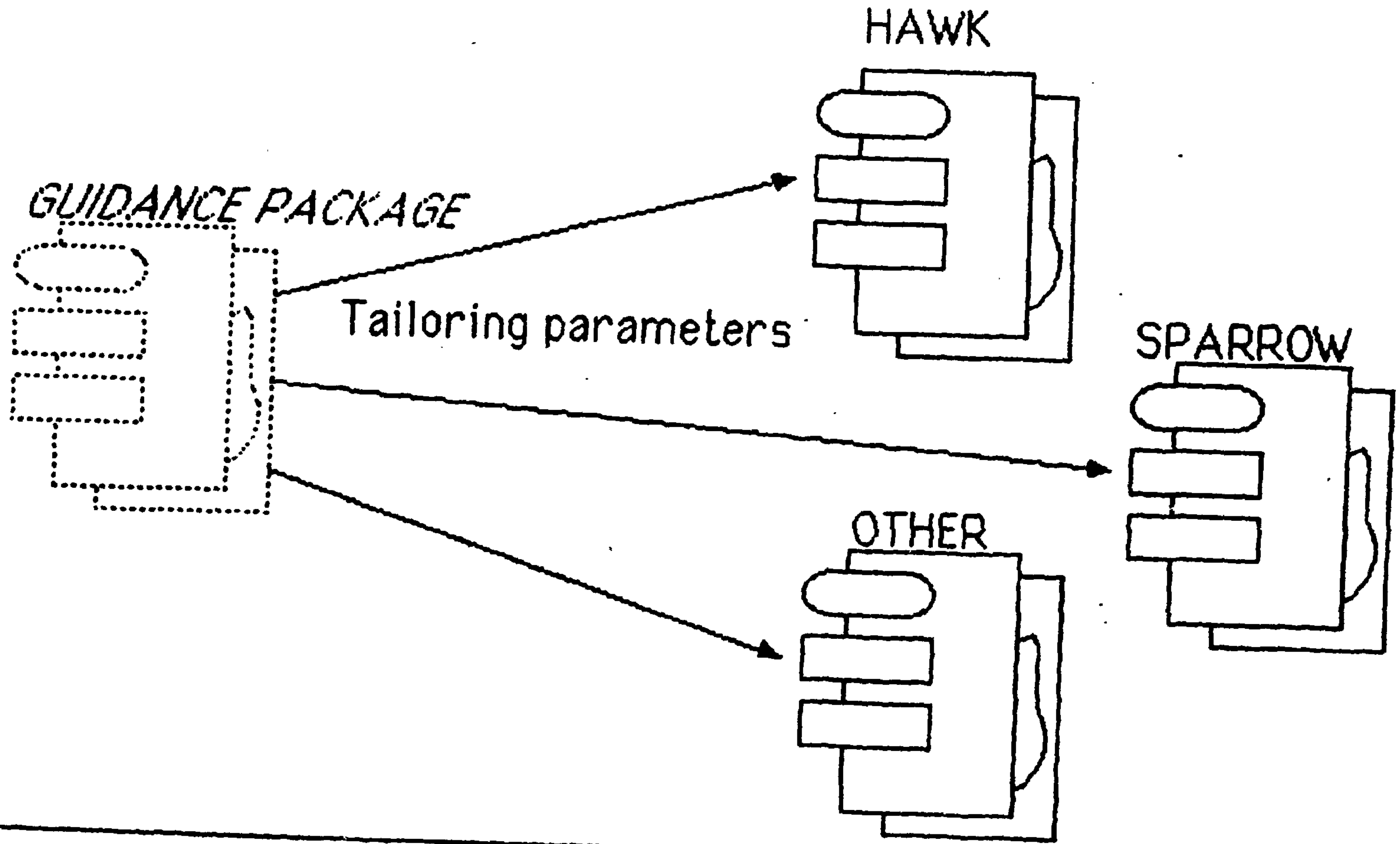
Major Features Of Ada

- Standardization
- Readability
- Program Units
- Separate Compilation
- Subprograms
- Packages
- Strong Typing
- Typing Structures
- Data Abstraction
- Tasks
- Exceptions
- Generics

Generics

A generic is a tailorable template for a program unit

Increases ability to define reuseable software components



Generics

Reduce size of program text

Reduce need to reinvent the wheel

Increase reliability by allowing reuse of known components

Overview

Rationale for development

Capabilities and advantages

$\boxed{\text{Life Cycle application}}$

Software Life Cycle

Requirements analysis

Preliminary design

Detailed design

Coding and unit testing

CSC integration and testing

CSCI testing

Maintenance

Requirements analysis

Standardization brings a much higher level of predictability

- Ada language itself
- Existing Ada software components

Ada supports rapid prototyping very well

Design

Ada features support architectural design

Can actually express design in terms of PDL
- Compileable
- Allows other automated tool support

Can enforce design through compileable PDL

Ada supports varied methodologies

Ada features reduce need to squeeze design into a programming language

Coding

Ada features ensure original design is not violated

Using PDL reduces amount of coding activity

Readability of Ada code promotes productivity

Testing

The ability of Ada to support independent components allows more effective testing

Exceptions allow "built-in" testing facilities

Integration and testing

Ada PDL ensures interfaces are correct

More effective time can be spent testing the system rather than fixing integration errors

Maintenance

Readability makes maintenance much easier

Proper software engineering using Ada will reduce maintenance costs